

Colored Visual Tags: a Robust Approach for Augmented Reality

Andrea Dell'Acqua[§], Marco Ferrari[◊], Marco Marcon[§], Augusto Sarti[§], Stefano Tubaro[§]

[§]Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza L. da Vinci, 32, 20133 Milano (Italy)

[◊]CNR - IEIIT at Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza L. da Vinci, 32, 20133 Milano (Italy)

Abstract

This paper presents a robust method for fast Visual Tags reading, suitable for Augmented Reality (AR) environments. Tag detection is based on well known tools of image-processing, but their combination, together with the use of colored markers, allows a robust recognition even with low-cost CMOS or CCD cameras and in poorly illuminated environments. In particular the color mix and the structure of the tag are quite unusual in common environments and can be easily detected with color filtering and geometric analysis. The proposed tag carries binary information encoded in its structure: in the presented implementation a 32-bit code with 12 parity bits is encoded in the tag but extensions to longer codes can be easily devised.

1. Introduction

In Augmented Reality environments synthetical augmentations are attached to views of the real world. AR systems often need links between physical objects and digital information regarding them. A wide range of applications derives from such systems, providing the user with additional information or activating pre-determined actions.

In order to reliably recognize real objects, each object is assigned an ID, encoded in a tag. Visual Markers derived from barcodes are usually employed [5, 6, 7, 8, 9], but also Infra-Red (IR) or Radio Frequency (RF) tags have been proposed [3, 4]. A tagging technology turns out to be more suitable than another depending on the characteristics of the tagged objects and of the global environment.

In this paper we focus on the Visual Tag approach, which has several advantages: tags do not require batteries, can be printed by commercial color printers, and can be easily attached on books, badges, and so on. Moreover, such technology is low cost and does not require further equipment to the user: a commercial webcam is the only sensor required, which is already part of the normal equipment of an AR system user.

The traditional Visual Tag detection algorithms are based

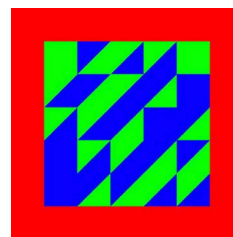


Figure 1: An example of a colored target used in our algorithm.

on gray-scale images and use black-and-white tags [7, 8, 9]. Their reliability in detection and decoding depends often on the illumination conditions. Usually, grayscale image are binarized [9], or an adaptive thresholding algorithm is applied to the luminance histogram in order to identify grey levels of the original black-and-white tag [7]. However, for this approach to be effective, only the histogram of the marker's area (and not of the whole image) should be considered, thus the marker should already be detected. Moreover, when the color information is discarded, other objects present in the scene may resemble the shape of the marker and misleading tags may be detected.

Today color webcams are available at very affordable prices, thus in several applications colored marker can be employed, giving robustness to the whole recognition process.

If the markers are strongly characterized in the color space, it is easier to detect them in a complex scene; moreover, an analysis of the colors in the HSV space turns out to be very effective in the presence of poor illumination conditions.

The tag we propose is a 5x5 grid of squared blocks divided diagonally, resulting in 50 triangles which represent the bits of the code (an example is shown in fig. 1).

Each triangle is green or blue depending on the binary value of the represented bit; the *code-area* of the marker is bounded by a red border.

Three blocks of the grid (located in three of the four corners of the code-area) are used to recover the correct orientation of the marker, thus a string of 44 bits can be encoded in the marker. The marker structure that we propose has been inspired by [9].

Since inaccuracy in the marker localization may lead to errors in the decoding step, the number of information bits in the marker has been reduced in order to implement an error-correction strategy, as suggested in [9].

Thus a string of 32 bits is encoded in the code-area of the marker, and the remaining 12 locations are used to code parity bits.

The error-correction algorithm we chose is based on a *soft* technique, described in Section 4.

The algorithm proposed in this paper can be mainly summarized in the following three steps:

1. Marker detection.
2. Extraction of the code from the marker.
3. Code reading and correction.

In the following sections each step is described in more detail.

2 Marker Detection

The first step of the algorithm consists in a fast and rough analysis of the input image looking for the presence of possible markers. A good algorithm should be able to find a badly illuminated or deformed marker, but also should not identify as markers misleading regions in the image.

In order to fulfil such requirements, our algorithm selects at the beginning a considerable number of possible markers, extracted by scanning the image. Refinement steps are then applied to the obtained regions using a cascade approach that progressively eliminates false recognitions.

As a first step, the HSV representation of the image is considered: a blob expansion algorithm is run on each pixel which exhibits a considerable saturation value and a hue value close to blue or green. Thresholds in this step are not to be very selective, in order to allow the algorithm to detect markers in badly illuminated scenes, where colors are not saturated and hue values may vary significantly due to color noise introduced by low-cost cameras. All the blobs that are compatible with the chosen thresholds are selected as candidates.

In the next step the statistical distributions of the red and blue/green (together) pixels are considered for each candidate.

Starting from the center of the bounding box of the blob, a discrete number of rays are considered in each direction and the two spatial distributions are computed (we consider

36 directions, i.e. one every 10 degrees). The integration along each ray is stopped when a non-red pixels is found after the red border.

Then the baricentra of the red and of the blue/green distributions, along with their variances are computed and normalized to the length of the radius, and finally their ratio is evaluated.

Target candidates detected in the first stage are tested basing on such ratio, which roughly characterizes the relative position of the red border and the green/blue area.

Then, a further analysis on the geometry of the selected candidates is carried out:

1. A set of pixels lying on the boundary of the marker are extracted (one for each ray considered in the previous step). Such points, if the candidate is a real marker, must lie on four lines corresponding to the marker sides.
2. A Hough transform [2] is computed on these points.
3. The four most relevant maxima in the Hough space are extracted: they correspond to the four '*maximum consensus*' lines for the set of considered points.
4. Such lines are ordered and intersected in order to find an estimate of the four corners of the marker.
5. The length of each marker side is then computed, and the ratio between the lengths of adjacent sides provides another criterion for discarding spurious or unreadable markers.

As a matter of fact, we are interested in detecting only markers resembling a (deformed) squared shape. If the image of the marker is too deformed by perspective, or even if the candidate is neither a marker nor a four-sided polygon, the lengths of two adjacent sides can be very different.

We suggest to accept as real markers only candidates which exhibit the four ratios between adjacent sides comprised between 0.5 and 2. A marker candidate which has passed all the checks described above has a very high probability of being a real marker.

A last refinement in the target definition is obtained with an accurate corner detector applied in a small region around the corners previously obtained from the intersection of '*maximum consensus*' lines.

3 Extraction of the code from the marker

Once a marker has been detected, its code can be retrieved.

The common way to proceed is to read the code from a previously rectified marker, i.e. from an image of the

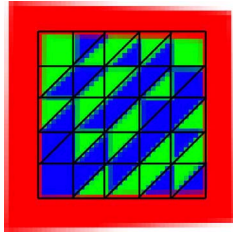


Figure 2: A rectified and normalized image of a target: misalignments between marker and template are present, but the code can be exactly retrieved

marker where projective deformation has been removed. A planar marker in the scene and its projection on the camera plane are related by a homography (or projectivity) relation [1]. Since the homography parameters can be recovered by considering four points and their transformed version, the marker can be rectified by applying to its image the transformation that maps its four corners into the vertices of a square.

It is also possible to perform a check on the rectified marker, in order to discard spurious markers accepted during the previous tests: the marker can be compared with a template of the same size, and the grade of correlation can be used as a merit function to discard false recognitions¹. After this check the marker undergoes a normalization process. Firstly all the pixels of the code-area are thresholded in the hue plane in order to assign them the values of green (corresponding to the binary value of 0) or blue (1).

Then, the four squares (see fig.1) at the corners of the code-area are considered in order to find the correct orientation of the marker: we adopted the convention that the two squares on the top corners are green while the square on the left-bottom corner is blue. The normalized marker can now be decoded using the template mask which is shown in fig. 2.

If the localization of the marker is inaccurate, misalignments may arise between the target and the reading template. Sometimes remarkable misalignments, together with the color noise introduced by the camera, can lead to decoding errors: in order to minimize this risk we adopted a code-correction strategy. We assign a value to each triangle corresponding to the average of its pixels: in particular if all pixels inside a template triangle are green (blue) the assigned value will be 0 (1). However, in the presence of misalignments and noise, pixels of adjacent triangles will corrupt the estimate, thus leading to a non-integer

¹Green and blue pixel in the code-area are obviously regarded as the same for the correlation computation, since the code changes from a marker to another.



Figure 3: A tag detection in a real scene: the IP address of the printer is encoded in the tag.

value comprised between 0 and 1. A classical two-levels code-correction technique could be employed at this stage, by setting a threshold and rounding the value of each triangle to 0 or 1. However a soft approach is more suited to our case, since it exploits all the available information. Two-levels code-correction techniques are usually devised to correct spot errors (a small number of wrong bits in a long string), while misalignments errors between reading mask and rectified marker usually affect the value of most of the bits of the string (as shown in fig. 2).

4 The Marker Error Correcting Code

A forward error correction scheme with a block length $N=44$ adapted to the geometrical structure of the tag has been considered; the code contains an information stream of size $K=32$. To simplify code design we do not try to exploit possible two-dimensional spatial correlation of the errors. The connection between the true bits encoded in the considered tag and those detected by the analysis of the acquired images can be seen as the transmission of the tag bits over a noisy memoryless transmission channel. Classical algebraic block codes are used on memoryless channels to correct errors at the receiver (see a.e. [10]).

Fig. 4 shows the typical performance curves of a set of FEC schemes that we have considered. Bit and Word probability of error (BER and WER) are shown as functions of E_b/N_0 , the typical parameter of transmission channels. In particular E_b is the energy per information bit and $N_0/2$ is the variance of the additive gaussian noise that can approximate quite well the distribution of the wrong pixels in our

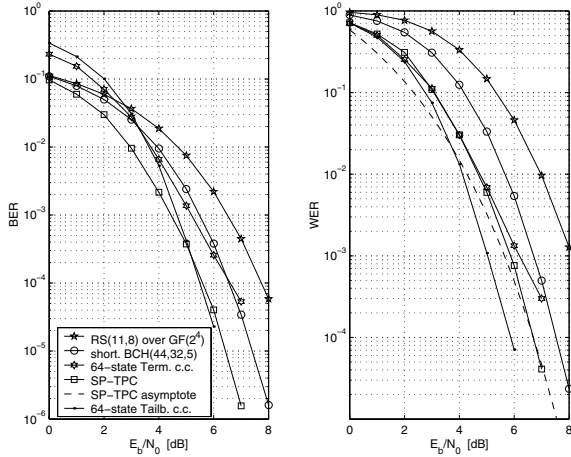


Figure 4: Performance of various error correcting codes.

readings.

A double-error correcting binary code which meets our needs is the BCH(63,51,5), which can be shortened to (44,32). This code can recover any pair of errors in the codeword. A non binary Reed-Solomon code requires a Galois Field of size at least 16, and provides only 3 bytes of redundancy hence it can correct one single byte error. Its performance is therefore pretty disappointing (Fig. 4).

As shown above, the nature of the marker detector allows to feed the decoder with soft (i.e. probabilistic) estimates of the bits, as example Log-Likelihood-Ratios (LLRs), which are not used by algebraic block decoders. It is well known that the knowledge of this information increases the capacity of the channel and improves the decoder performance. To exploit this advantage we could use a convolutional code; in this case the Viterbi algorithm using soft estimates (as example LLRs of the bits) performs Maximum Likelihood (ML) decoding. More recently developed error correcting schemes such as Turbo Codes, Turbo Product Codes (TPC) or the renewed LDPC codes, use concatenated codes and iterative soft decoders which perform almost as ML or Maximum A Posteriori (MAP) decoding, with lower complexity.

Applying a convolutional code we could either use a very high rate (7/8) terminated code, or a medium high rate (3/4) tailbiting code. A little puncturing correction is needed in both cases to meet the code constraints. In both cases we incur in an original use of convolutional codes, which are in general designed for long continuous time information streams. The second choice enables with 64 states to keep a nominal minimum hamming distance of 5 as the BCH(63,51) code, but the advantage of soft decoding gives almost 2 dB gain over the BCH code. Of course this is paid with an increased complexity of the receiver: the Viterbi

algorithm in this case requires almost 1000 additions and comparisons per information bit.

A low complexity solution based on iterative decoding requires the concatenation of two or more codes and the iterative exchange of soft information between the corresponding Soft-In-Soft-Out (SISO) decoders. The use of extrinsic information helps the convergence of the decoding process where each decoder uses the information coming from the others as a priori knowledge on the symbols. The simplest case of parity code and SISO decoder are given by Single Parity Check (SPC) codes. When a SPC code is imposed on a n-tuple, the extrinsic posterior LLR of the i^{th} bit is given by [11]:

$$L_e(b_i) = t^{-1} \left[\prod_{k=1, k \neq i}^n t(L(b_k)) \right] \simeq \left[\prod_{k=1, k \neq i}^n \text{sign}(L(b_k)) \right] \min_{k \neq i} |L(b_k)| \quad (1)$$

where $L(b_k)$ are the LLRs of the received symbols (plus "a priori" information, when available), and $t(x) = \tanh(x/2)$.

Due to the very small block size random or random like interleaving does not provide any meaningful advantage, hence we use two-dimensional parallel-like product SPC codes (SP-TPC): we arrange the bits into an 8×4 matrix and we add an SPC check to each row and each column. At each iteration the decoder performs a horizontal decoding based on 1 using $L(b_k)$ and $L_e^{(V)}(b_k)$ as "a priori" information when available, and a vertical decoding using $L(b_k)$ and $L_e^{(H)}(b_k)$ when available. This second step also provides an a posteriori LLR of each information bit given by:

$$L_P(b_k) = L(b_k) + L_e^{(H)}(b_k) + L_e^{(V)}(b_k) \quad (2)$$

The structure of the code enables very simple and fast parallel wise horizontal and vertical SISO decoding. The convergence of the decoding process is always guaranteed [12], and it practically occurs after two or three iterations. After convergence, the approximate posterior LLR (2) is very close to MAP decoding.

The bit and frame error probability of this code are plotted in Fig. 4 after three iterations, using the approximation of (1). Note that this scheme performs only 0.5 dB worse than the tailbiting 64 states Viterbi, with only 6 additions and comparisons per information bit.

On the basis of this result we chose to adopt this coding algorithm in our application providing a robust correction code to the tag reading routine. Experimental simulations on synthetic and real marker images have shown a Bit Error

Rate and Word Error Rate consistent with the theoretical values for SP-TPC codes shown in fig. 4.

5 Conclusions

In this paper a Tag Detection and Decoding system is proposed, based on visual (printed) markers.

Color information grants robustness to the whole system with respect to poor illumination conditions, and allows the system to work well even in the presence of low quality images acquired by cheap color webcams.

The particular selection of the colors also makes the tag localization possible even in complex scenes.

Furthermore, an iterative soft code-correction scheme is employed which gives robustness to the reading step.

In spite of all the checks run to discard spurious markers, the time required for detection, decoding and code-correction is about 0.5 seconds (tested on a PC equipped with a Pentium IV processor and 512MB of RAM), which is good for almost real-time applications.

We think that our algorithm can be proposed as a good candidate for demanding AR applications. In fig. 3 we show a possible application: the IP address of a network printer is encoded in a Visual Tag. Once the tag is detected and decoded, an AR user equipped with a laptop connected to a wireless network can print documents on the printer.

Acknowledgments

The work presented was developed with the cooperation of FIRB-VICOM, a project funded by the Italian Ministry of University and Scientific Research (MIUR).

References

- [1] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.
- [2] F. O’Gorman and M. B. Clowes. Finding Picture Edges Through Collinearity of Feature Points, *IEEE Trans. on Computers* 25(4), Apr. 1976, pp. 449-456.
- [3] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, January 1992.
- [4] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI’99 Proceedings*, pp. 370-377, 1999.
- [5] B. M. Lange, M. A. Jones, J. L. Meyers. Insight Lab: An immersive team environment linking paper, displays, and data. In *CHI’98 Proceedings*, pp. 550-557, 1998.
- [6] N. Khotake, J. Rekimoto, and Y. Anzai. InfoStick: An interaction device for inter-appliance computing. In *Workshop on Handheld and Ubiquitous Computing (HUC’99)*, 1999.
- [7] ARToolKit: http://www.hitl.washington.edu/research/shared_space/download/
- [8] J. Rekimoto, Y. Ayatsuka, "CyberCode: designing augmented reality environments with visual tags, *Proceedings of DARE 2000 (Designing Augmented Reality Environments)*, Elsinore, Denmark, 12-14 April 2000, pp. 1-10.
- [9] T. Kawano, Y. Ban, K. Uehara. A Coded Visual Marker for Video Tracking System Based on Structured Image Analysis. *ISMAR 2003*: 262-263.
- [10] Sloane, Mc Williams, *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977
- [11] J. Hagenauer, E. Offer, L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429-445, Mar. 1996
- [12] M. Ferrari, S. Bellini, "Existence and uniqueness of the solution for turbo decoding of parallel concatenated Single Parity Check codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 722-726, Mar. 2003